BIOSTATS 640 – Intermediate Biostatistics
Spring 2022

Introduction to R
02 – Introduction to Packages and Simple Data Description

On to real data! In this lesson, we will do our first import of excel data and produce some descriptives using commands in packages. But first, we will learn how to work with R Script files.

__1. Highlights from Lesson 01 – R Essentials
__2. How to Work with R Script Files
__3. Introduction to Packages (e.g., tidyverse, summarytools and ggplot2)
__4. Import Excel Data
__5. Produce some numerical descriptives using the package {summarytools}
__6. Produce some graphs using the package {ggplot2}

Introduction to the arthritis dataset.

Citation:
Edward Gracely, "Arthritis Treatment Dataset", *TSHS Resources Portal (2020)*. Available: https://www.causeweb.org/tshs/arthritis-treatment/.

This was a study of treatment for rheumatoid arthritis (RA) in elderly patients. With the availability of effective anti-RA agents, disease activity measurements can inform the specification of treatment and are part of an approach known as "treat to target". Of interest in this study was the possibility that, compared to their younger counterparts, older patients with RA might be less likely to have their disease activity measured and less likely to receive aggressive treatment. The study design is retrospective cohort study that compared two group (control patients age 40-70 years versus elderly patients age 75+ years) with respect to outcomes measuring disease activity. The sample size is 530 (459 controls + 71 elderly) and there are 14 variables.

We will be using the arthritis dataset in several lessons. In this lesson (02), we use this dataset to illustrate some basics of data description in R.

Partial Data Dictionary

Position	Variable	Variable Label	Units	Codes	Missing data
1	ID	Unique subject ID			None
2	Age	Age in years	Years	90 = 90+ for HIPAA purposes	None
3	AgeGp	Age Group		1 = 40 to 70 years ("control") 2 = 75 and older ("elderly")	None
4	Sex	Patient sex		0 = Female 1 = Male	None

#1. Highlights from Lesson 01
R Essentials

1. **Commands are executed from the console pane**
2. **The prompt is >**
3. **The assignment operator is <-**
4. **The console can be used as a giant calculator.**
5. **Commands in the console pane are *NOT saved***
6. **+ in the console pane is telling you that your command is INCOMPLETE.**
   Two solutions are possible, either: (1) finish it (type remainder of command at the +) ; or (2) abandon it (escape)
7. **Use the UP ARROW key to retrieve, edit, and re-run earlier commands**
8. **< control > l (letter el) clears the console pane display**
   No worries. Your history is NOT lost.
9. **Use the function c( ) to create a vector object**
   Example: Create vector object v1 (left hand side) and assign (<-) to it the vector (1,2,3,4) (right hand side)
   v1 <- c(1,2,3,4)
10. **Statistical functions run on COMPLETE data only. Tip: use option na.rm=T to remove missings (NA)**
    Example:
    ```
    mean(v1, na.rm=T)
    ```
    You could also do the following:
    ```
    mean(v1, na.rm=TRUE)
    ```

#2. How to Work with R Script Files

In Lesson 01 we did all our work in the console
Think of the console as the engine.
From here, commands are submitted and R executes them.
Commands in the console are NOT saved.

In this lesson (02) we will type our commands into a saved R Script
An R Script file is a plain text file
Commands get sent to the console to be executed
Commands in An R Script file can be SAVED.
Tip. Create saved R Script files that are permanent records of your analysis code.

In the next lession (03) we will learn how to produce archived work in an R Markdown file
An R Markdown file is a specialised type of file that contains: text, R code, and a utility that produces output (pdf, html, etc)
Tip. Create saved R Markdown files that are permanent (and reproducible) records of your analyis and resulting output.

Summary: Console v R Script v R Markdown

	Console	R Script	R Markdown
At a Glance	**Calculate**	**Save your code**	**Archive all your work**
What is it	R Studio interface. Here, commands are submitted directly to R and are executed immediately.	Text editor + utility that sends commands to the console where they will be executed	Text editor + Embedded R commands + utility that sends commands to the console where they will be executed + utility that produces (renders) a report in any of several formats (e.g, html, PDF, Word)

What is an R Script file?
It is a plain text file that contains R commands and comments.
R Script files allow you to create a permanent record of your work that can be archived or shared.
R Script files can be shared because they are text files.

Tips. Console versus R Script: *Always do your work from an R Script file (or an R Markdown)!*

- Do not regularly work from the console pane
- Use the console pane for just TWO THINGS ONLY: (1) as a calculator and/or (2) when requesting help
- Make it a habit to do all your programming work in a saved R Script file (unless you are creating an R Markdown file)

Suggestions for Writing R Script Files

- When you create a new R script file, save it immediately to a permanent file
- Save often.
- Use lots of comments (remember: these begin with the hashtag/pound symbol #)
- Begin your R script file with a structured set of comments that describe the R Script file (title, date, name of file, etc)
- Include the following as the first command in your R Script file: **rm(list=ls( ))**

Tip. At the start of your session, clear your workspace
"clear R's brain", "remove all objects from the environment", "clear the decks"

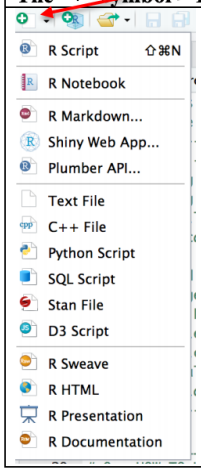
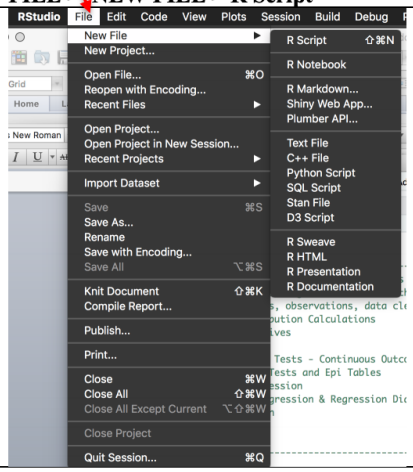
```
rm(list=ls( ))
```

- How this works: **ls( )** tells R "get every object in my workspace". **rm( )** tells R to remove it.

- Why this is a good idea: You don't want to mistakenly work with objects leftover from a previous session.

How to Create an R Script File

STEP 1: From the source pane, click on either (+) or do FILE > NEW FILE > R SCRIPT

The "+" symbol > R Script	FILE > NEW FILE > R Script
	

STEP 2: Immediately save it using **FILE > SAVE AS.**

How to execute commands in your R Script file

So. You've typed some commands into your R Script file. Now you want to execute some or all of them!
Remember: What you are actually doing is sending your commands to the console to be executed there.

Method 1 – Execute selected commands only.
Highlight commands → At upper right, click on **RUN**

Method 2 – Execute selected commands only.
Highlight commands → Do a **<control>-enter**

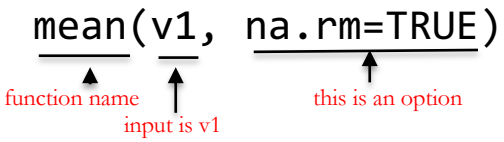
Method 3 – Execute entire R Script file.
At upper right, click on **SOURCE**

All done?
Don't forget to save your R Script file before exiting R Studio

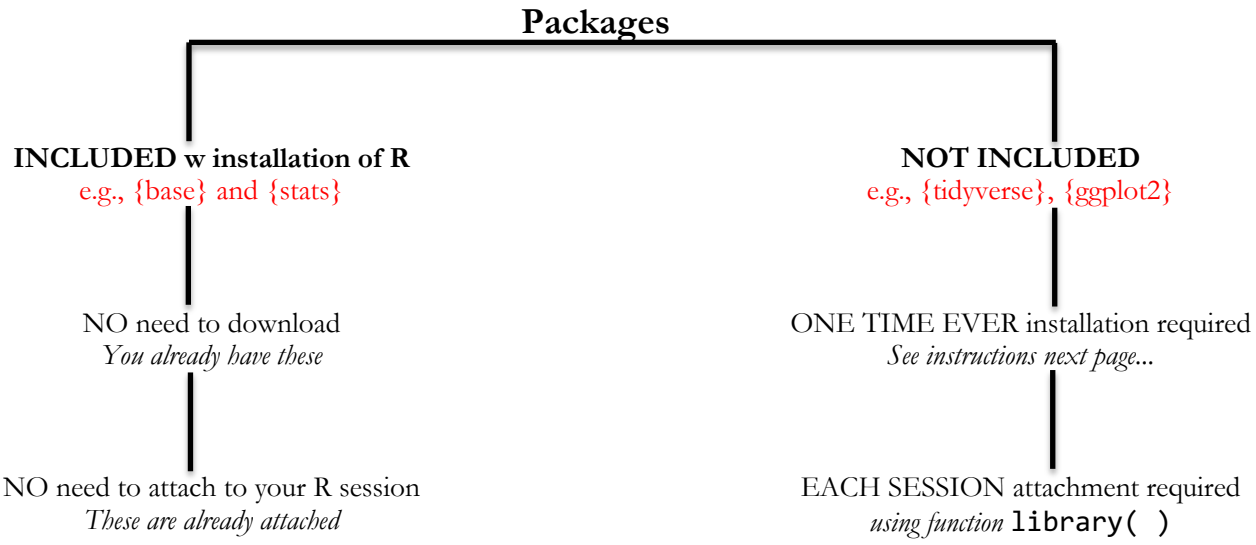
#3. Introduction to Packages

Good to Know!

- When it really comes down to it, for the most part, our work in R is just lots and lots of calls to **functions**.
- **What is a function?** A function is some pre-existing,"ready to go", executable code that someone else has written for you (*you're welcome!*). You simply call (invoke) it. A function has an opening parenthesis "(" and a closing parenthesis ")" In the parentheses are the inputs (also called arguments) and options that you supply, separated by commas. How it works: You invoke/call the function. R then executes that function (performs some tasks) using the inputs an options you provide. The output/results are then passed on. The output may be shown to you on your screen. The results may also be passed on as input to other functions. `mean( )` is an example of a function. Here is a closer look at its structure:



- A **package** is simply a collection of functions (again, these have been previously programmed for you. Nice!
- There are two (2) types of packages: 1) included: packages that are included in the installation of R; and 2) not included: packages that are not included in your installation of R. How you work with packages is slightly different, depending on the type:



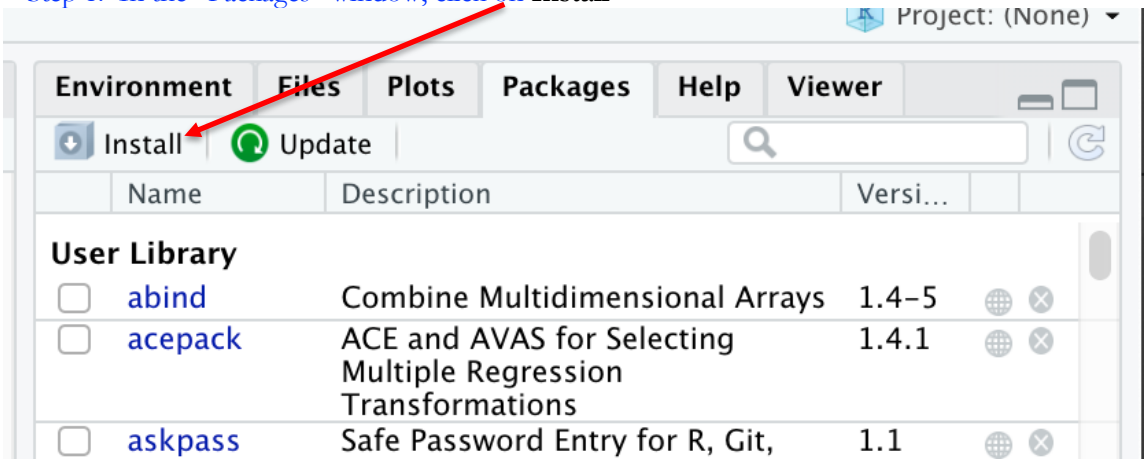
How to Install a Package
Remember: Installing a package is done just one time. Ever.

There are two ways to install a package:

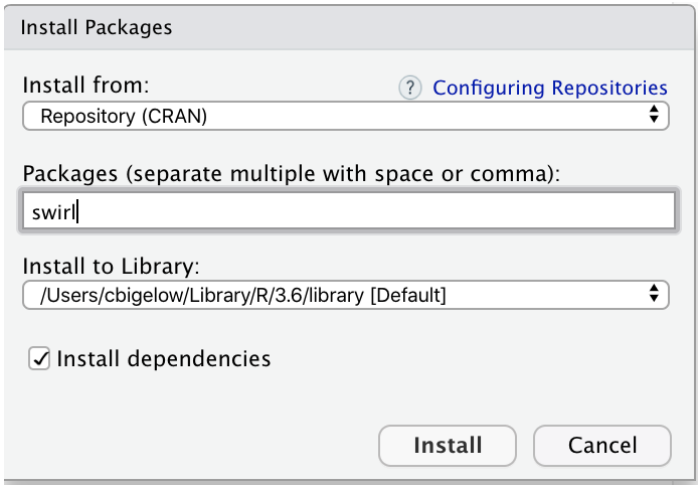
1. Method I (Menu Driven) – Using the drop down menus in R Studio
2. Method II (Direct Command) – In the console pane, using the command `install.packages(" ")`

Method I: Menu Driven (Recommended)

Step 1: In the "Packages" window, click on **Install**



Step 2: In the "Install Packages" window
Example: I want to install the package called swirl
In Install from: **default (Repository CRAN) is fine**
In Packages (separate multiple with space or comma:) **type in name of package (e.g. swirl)**
In Install to Library: **leave as is**
Check box for "Install dependencies": **check this**
At bottom, click **Install**



Step 3: Be patient. Wait. Then take a look at your console window. You should see:
At bottom you should see the prompt ">".

```
[Workspace loaded from ~/Desktop/.RData]

> install.packages("swirl")
Installing package into '/Users/cbigelow/Library/R/3.6/library'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/swirl_2.4.5.tgz'
Content type 'application/x-gzip' length 348039 bytes (339 KB)
==================================================
downloaded 339 KB


The downloaded binary packages are in
        /var/folders/rn/drwz6qbx3nb0ycr7vlppztt52nq3lw/T//RtmpSV27qs/downloaded_packages
>
```

Method II: Direct Command

In the console window Command Driven (From the console window), issue the command `install.packages( ).` Here, the name of the package IS ENCLOSED IN QUOTES.

For example:
`install.packages("ggplot2")` Don't forget: package name IS ENCLOSED in quotes

How to Attach a Package to Your R Session
Remember: To use a package in your session, you must tell R that you want it attached.
To do this issue the command `library( )`.
Here, the name of the package is NOT enclosed in quotes

For example:
`library(ggplot2)` Note: package name is NOT enclosed in quotes

Your Turn.
Install the following 3 packages

1. summarytools
2. ggplot2
3. tidyverse

What Could Go Wrong

#1. You forgot to do the "one time" installation of that package.

#2. You have installed the package (hooray) but you forgot to issue the command `library( )` to attach that package

#3. *Really annoying- Your command will not execute!.* Sometimes the command you are using has the same name in more than one package and you happen to have attached those packages to the same R Studio session. R Studio gets confused.

For example, if you want to use the command **`freq( )`** in the package **summarytools**

ERROR: And the following does not work

```
freq(framingham$sex)
```

SOLUTION: Tell R Studio which `freq( )` command you actually want to use by telling it the package
To do this, prefact your call to `freq( )` with the name of the package followed by a double colon
For example:

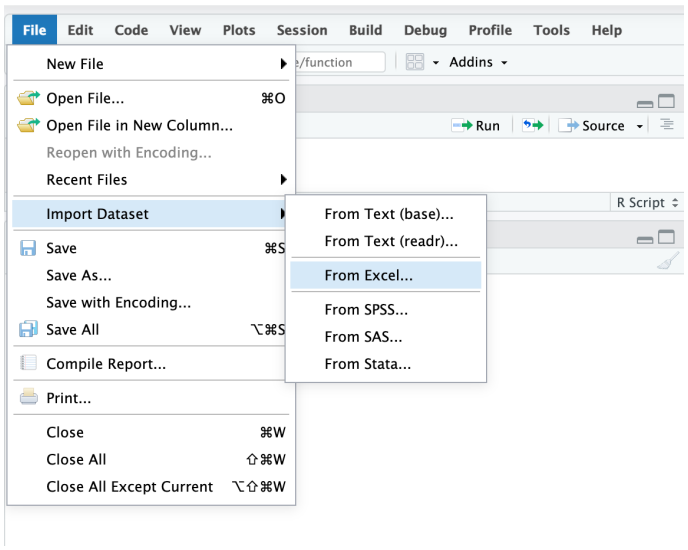
```
summarytools::freq(framingham$sex)
```

#4. Import Excel Data

There are multiple ways to import excel data. In this illustration, we use the drop down menus.

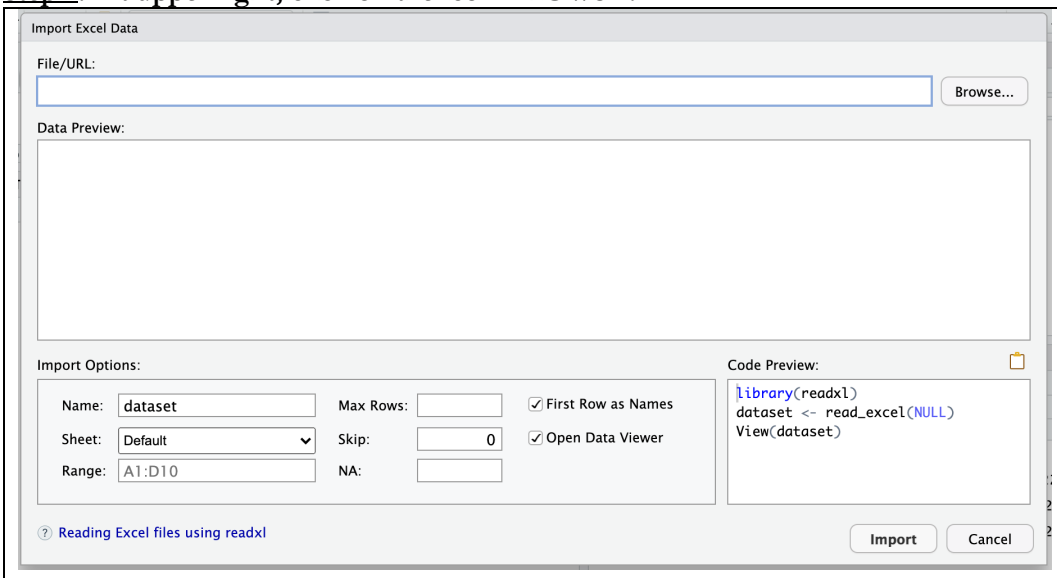
Before You Begin: Strongly encouraged:
(Source: marinstats lectures) Importing Excel Data into R (video, 8:12)

Step 1: At upper left; FILE > IMPORT DATASET > FROM EXCEL

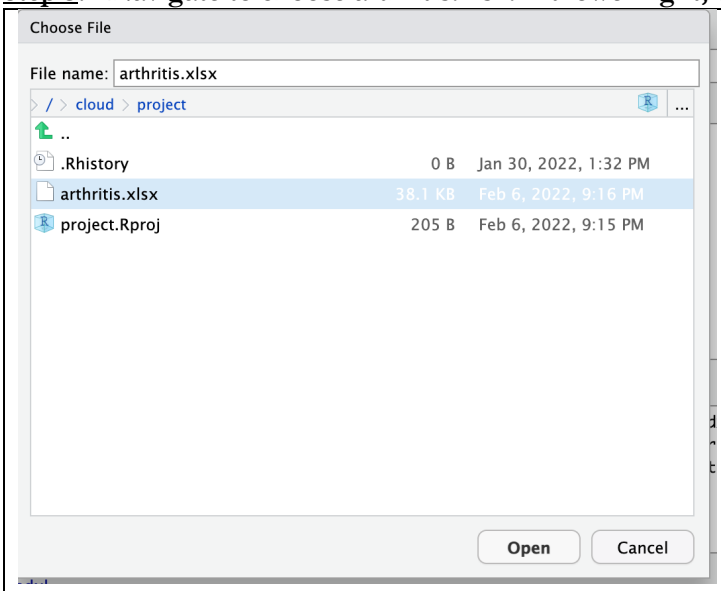


Note: R may return a message saying that you need to install readxl. Click YES. Then wait until you get a prompt.

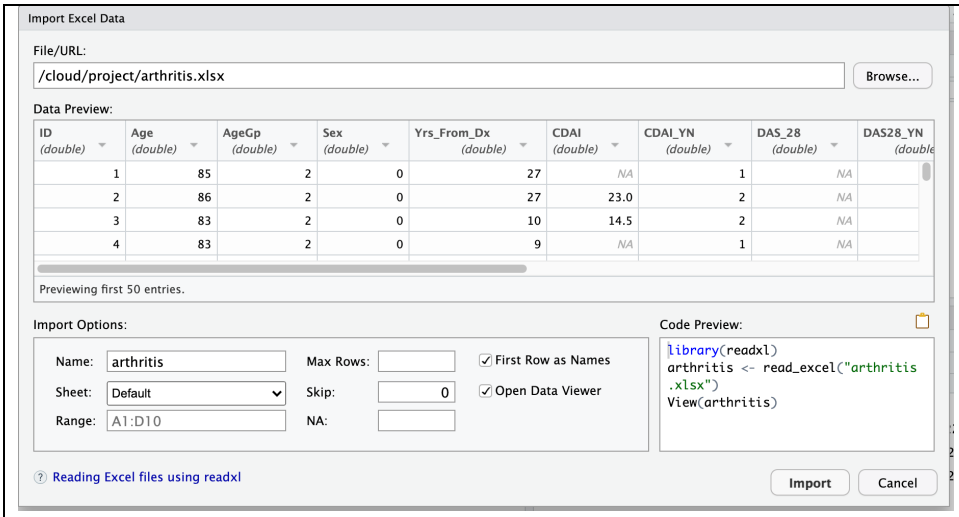
Step 2: At upper right, click on the icon BROWSE.



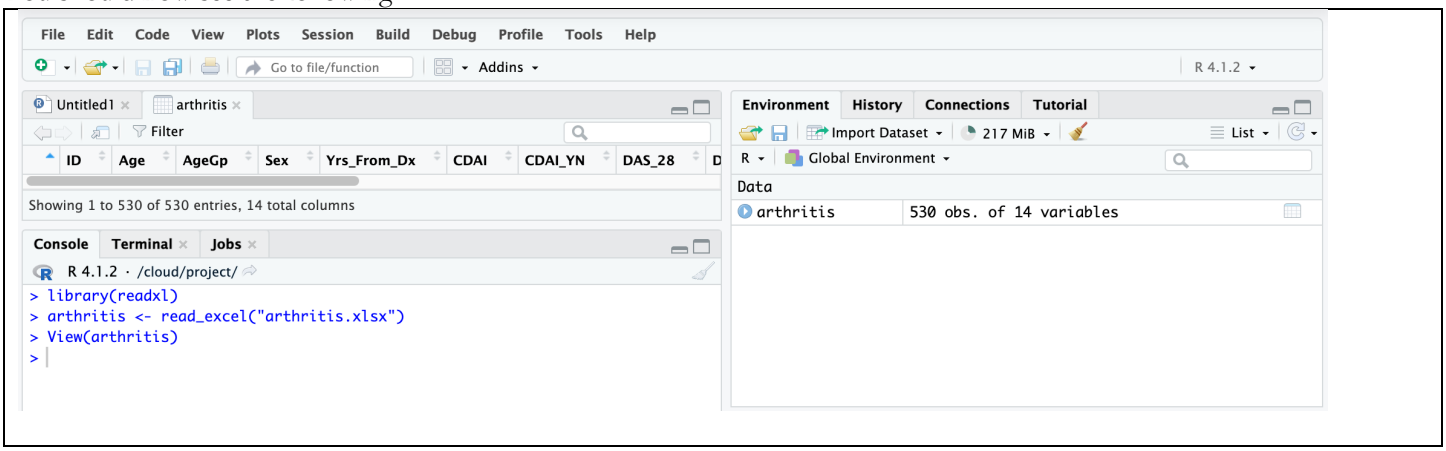
Step 3: Navigate to choose arthritis.xlsx. At lower right, click OPEN



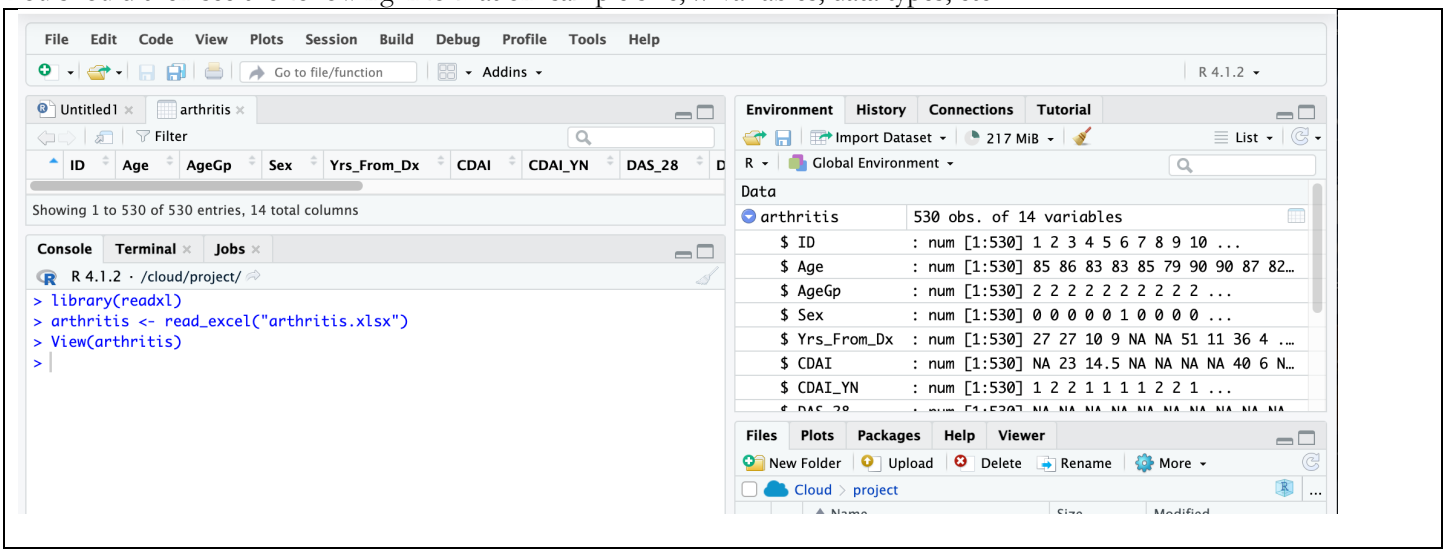
Step 4: Take your time here in making your selections. All set? At lower right, click IMPORT



You should now see the following



Step 5: At right, in the Environment tab, click on the down arrow next to the dataset name arthritis.
You should then see the following information: sample size, # variables, data types, etc.



#5 Produce Some Numerical Descriptives Using Package {summarytools}

```
# Illustration:  Numerical Summaries
# Step 1:  Attach package {summarytools}
library(summarytools)

# Inspect data using structure function str( )
str(arthritis)
```

```
tibble [530 × 14] (S3: tbl_df/tbl/data.frame)
 $ ID           : num [1:530] 1 2 3 4 5 6 7 8 9 10 ...
 $ Age          : num [1:530] 85 86 83 83 85 79 90 90 87 82 ...
 $ AgeGp        : num [1:530] 2 2 2 2 2 2 2 2 2 2 ...
 $ Sex          : num [1:530] 0 0 0 0 0 1 0 0 0 0 ...
 $ Yrs_From_Dx  : num [1:530] 27 27 10 9 NA NA 51 11 36 4 ...
 $ CDAI         : num [1:530] NA 23 14.5 NA NA NA NA 40 6 NA ...
 $ CDAI_YN      : num [1:530] 1 2 2 1 1 1 1 2 2 1 ...
 $ DAS_28       : num [1:530] NA NA NA NA NA NA NA NA NA NA ...
 $ DAS28_YN     : num [1:530] 1 1 1 1 1 1 1 1 1 1 ...
 $ Steroids_GT_5: num [1:530] 0 1 1 1 0 0 0 1 0 0 ...
 $ DMARDs       : num [1:530] 1 1 1 1 0 0 1 0 0 1 ...
 $ Biologics    : num [1:530] 0 0 1 0 0 0 1 0 1 0 ...
 $ sDMARDS      : num [1:530] 0 0 0 0 0 0 0 0 0 0 ...
 $ OsteopScreen : num [1:530] 0 1 1 1 0 0 0 1 1 1 ...
```

```
# We see that every variable is numeric.
# For illustration, it would be nice to have 2 categorical variables (R calls these factors)
# So let's create these from AgeGp and Sex.  This requires two steps as shown below.

# Create a factor variable of AgeGp called AgeGpf
# Label its values
arthritis$AgeGpf <- factor(arthritis$AgeGp)                 # Step 1:  convert from numeric to factor
arthritis$AgeGpf <- factor(arthritis$AgeGpf,                # Step 2:  label discrete values
                           levels=c(1,2),
                           labels=c("Control(40-70 yrs)","Elderly (75+ yrs)"))

# Create a factor variable of Sex called Sexf
# Label its values
arthritis$Sexf <- factor(arthritis$Sex)                    # Step 1:  convert from numeric to factor
arthritis$Sexf <- factor(arthritis$Sexf,                   # Step 2:  label discrete values
                           levels=c(0,1),
                           labels=c("Female","Male"))
```

```
# freq( ) in package {summarytools} to produce descriptive statistics of ONE categorical variable
summarytools::freq(arthritis$AgeGpf)
```

```
Frequencies
arthritis$AgeGpf
Type: Factor

                         Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
----------------------- ------ --------- -------------- --------- --------------
      Control(40-70 yrs)    459     86.60          86.60     86.60          86.60
       Elderly (75+ yrs)     71     13.40         100.00     13.40         100.00
                    <NA>      0                               0.00         100.00
                   Total    530    100.00         100.00    100.00         100.00
```

```
# ctable( ) in package {summarytools} to produce a xtab of TWO categorical variables
summarytools::ctable(arthritis$Sexf,arthritis$AgeGpf,prop="r")
```

```
Cross-Tabulation, Row Proportions
Sexf * AgeGpf
Data Frame: arthritis

-------- -------- -------------------- ------------------- --------------
           AgeGpf   Control(40-70 yrs)   Elderly (75+ yrs)          Total
    Sexf
  Female                  366 (85.5%)          62 (14.5%)   428 (100.0%)
    Male                   93 (91.2%)           9 ( 8.8%)   102 (100.0%)
   Total                  459 (86.6%)          71 (13.4%)   530 (100.0%)
-------- -------- -------------------- ------------------- --------------
```

```
# descr( ) in package {summarytools} to produce descriptive statistics of ONE continuous variable
summarytools::descr(arthritis$Age)
```

```
Descriptive Statistics
arthritis$Age
N: 530

                      Age
----------------- --------
             Mean    60.69
          Std.Dev    10.47
              Min    42.00
               Q1    54.00
           Median    59.00
               Q3    66.00
              Max    90.00
              MAD     8.90
              IQR    12.00
               CV     0.17
         Skewness     0.76
      SE.Skewness     0.11
         Kurtosis     0.27
          N.Valid   530.00
        Pct.Valid   100.00
```

```
# descr( ) in package {summarytools} to produce descriptive statistics of ONE variable
# option stats=c( ) to choose statistics to report
# option transpose=TRUE to display output horizontally (instead of default vertical display)
summarytools::descr(arthritis$Age,
                    stats = c("n.valid","mean", "sd","min","q1", "med", "q3", "max"),
                    transpose = TRUE)
```

```
Descriptive Statistics
arthritis$Age
N: 530

            N.Valid    Mean   Std.Dev     Min      Q1   Median      Q3     Max
--------- --------- ------- --------- ------- ------- -------- ------- -------
      Age    530.00   60.69     10.47   42.00   54.00    59.00   66.00   90.00
```

#6 Produce Some Graphical Descriptives Using Package {ggplot2}

Good to Know.
A graph produced using ggplot is built "layer by layer". Once you get the hang of it, it's kind of fun!

Step 1: Attach the package {ggplot2} using the command `library( )`
library(ggplot2)

Step 2 *(optional, recommended):* Create each plot in its own R Markdown chunk

Step 3 *(optional, recommended):* Code your graph line by line, layering as you go.

First line: # Execute and correct as needed
```
data = dataframename
```

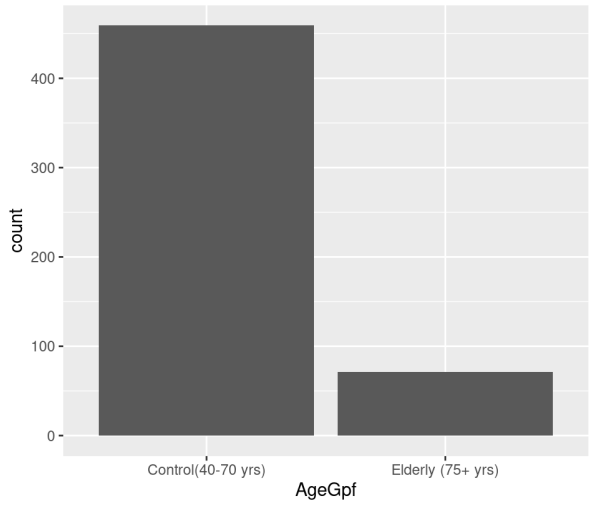
Second line is added # Execute and correct as needed
```
data = dataframename +
aes( )
```

Third line is added: # Execute and correct as needed
```
data = dataframename +
aes( ) +
geom_xxx( )
```

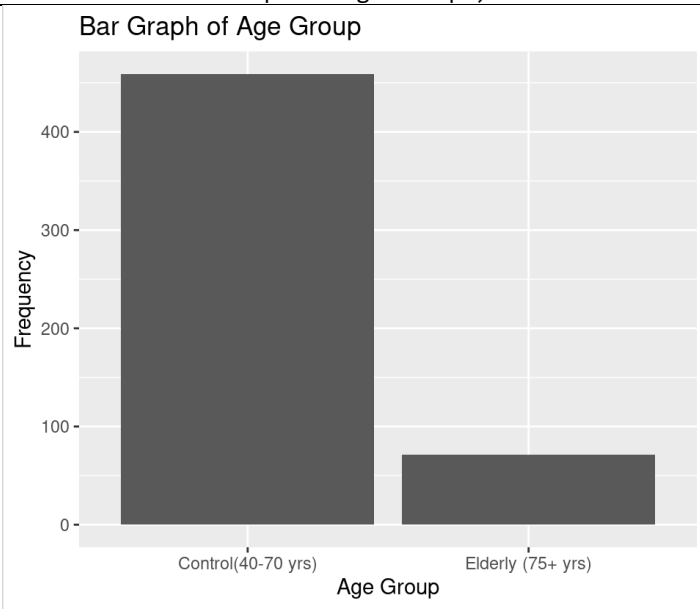
Tips

- The continuation + MUST BE at the end of the line (not at the start of the next line)
- As you add lines, execute all of the accumulating layers
- By building your graph layer by layer, it is easier to trouble shoot errors

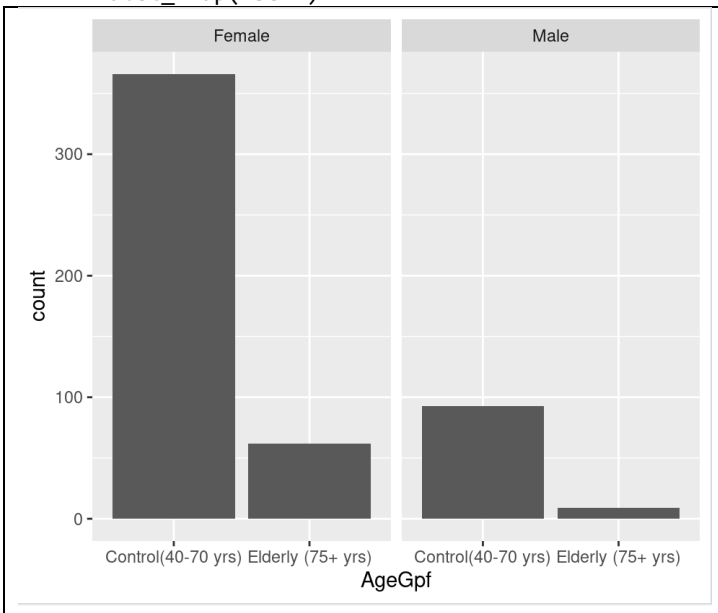
```
# Illustration:  Graphical Summaries
# Step 1:  Attach package {ggplot2}
library(ggplot2)

# ggplot( ) and geom_bar( ) in package {ggplot2} to produce Bar Graph of ONE categorical variable
ggplot(data=arthritis) +
  aes(x=AgeGpf) +
  geom_bar(na.rm=T)
```



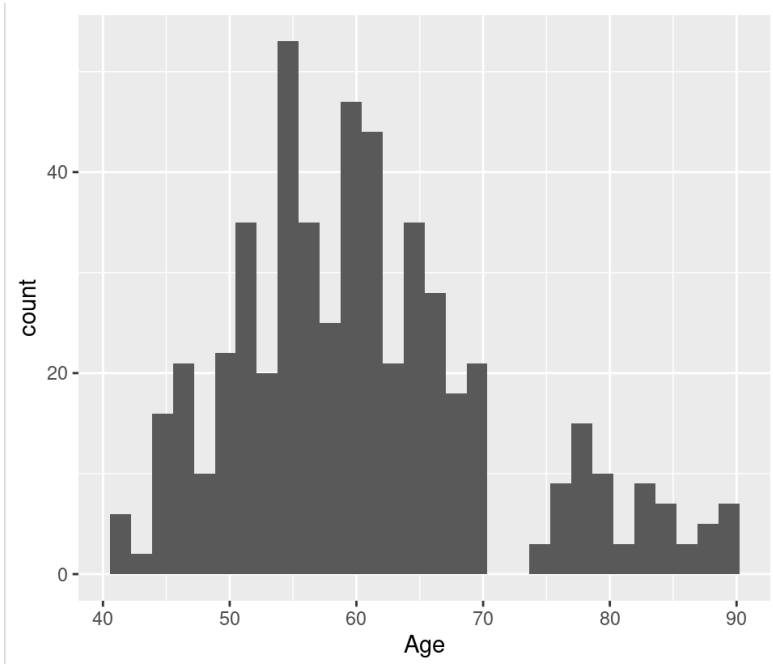

```
# ggplot( ) and geom_bar( ) in package {ggplot2} to produce Bar Graph of ONE categorical variable
# plus layer labs( ) to produce labels
ggplot(data=arthritis) +
  aes(x=AgeGpf) +
  geom_bar(na.rm=T) +
  labs(x="Age Group",
       y="Frequency",
       title="Bar Graph of Age Group")
```



```
# ggplot( ) and geom_bar( ) in package {ggplot2} to produce 2 way Bar Graph of TWO categorical variables
# using layer facet_wrap( ) to obtain bar graph of one variable, seperately for the levels of the other
ggplot(data=arthritis) +
       aes(x=AgeGpf)+
       geom_bar(na.rm=T)+
       facet_wrap(~Sexf)
```




```
# ggplot( ) and geom_histogram( ) in package {ggplot2} to produce Histogram of ONE continuous variable
ggplot(data=arthritis) +
       aes(x=Age)+
       geom_histogram(na.rm=T)
```



```
# ggplot( ) and geom_histogram( ) in package {ggplot2} to produce Histogram of ONE continuous variable
# with option binwidth to reset bin widths
# and later labs( ) to produced labels
ggplot(data=arthritis) +
      aes(x=Age,y=..count..) +
      geom_histogram(na.rm=T, binwidth=5) +
      labs(x="Age (yrs)",
           y="Frequency",
          title="Histogram of Age")
```